Measures and Measurement for Secure Software Development

Dave Zubrow, Software Engineering Institute [vita[3]]
James McCurley, Software Engineering Institute [vita[4]]
Carol Dekkers, Software Engineering Institute [vita[5]]

2005-09-28

This article discusses how measurement can be applied to software development processes and work products to monitor and improve the security characteristics of the software being developed. It is aimed at practitioners—designers, architects, requirements specialists, coders, testers, and managers—who desire guidance as to the best way to approach measurement for secure development. It does not address security measurements of system or network operations.

Overview

This practice area description discusses how measurement can be applied to software development processes and work products to monitor and improve the security characteristics of the software being developed. Measurement is highly dependent on aspects of the software development life cycle (SDLC), including policies, processes, and procedures that reflect (or not) security concerns. This topic area is aimed at practitioners—designers, architects, requirements specialists, coders, testers, and managers—who desire guidance as to the best way to approach measurement for secure development. It does not address security measurements of system or network operations.

Measurement and the Software Development Life Cycle

Measurement has long been recognized as a critical activity for successful system development. Good measurement practices and data enable realistic project planning, timely monitoring of project progress and status, identification of risks to the project, and effective process improvement. Measures and indicators of software work products such as requirements, designs, and source code can be analyzed to diagnose problems and identify solutions during project execution and enable the reduction of defects, waste (effort, resources, etc.), and cycle time. These practices enable organizations to achieve higher quality products and reflect mature processes, as delineated by the CMMI.[10] Watchfire has published a short description of typical application security activities for each level of the CMMI [Graf 05]. Unfortunately, measures for the development of securely coded products are in their infancy, and no consensus exists as to what measures constitute best practices. A review of the existing technical literature reveals the scarcity of *any* publicly reported, validated security measurements related to the software development life cycle [see Measurement - Business Case]. Nonetheless, there are some measures and practices used in software development that can be fruitfully extended to address security concerns.

SLDC areas related to the definition and use of measures for secure development addressed in the Build

3. daisy:224 (Zubrow, Dave)
4. daisy:225 (McCurley, James)
5. daisy:226 (Dekkers, Carol)
10. CMMI is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

Security In modules include: Requirements Analysis; Architectural Risk Analysis; Assembly, Integration and Evolution; Code Analysis; Risk-Based and Functional Security Testing; Software Development Life-Cycle Process; Coding Rules; Training & Awareness; and Project Management. Risk management in general is addressed separately in the module Risk Analysis Framework on the Build Security In website. In contrast to the traditional focus of risk management on project failure in software development, it must now be extended to address the malicious exploitation of product flaws. Threat modeling and its use in the SDLC is addressed in the modules Attack Patterns, Threat Modeling, and Historical Risks. All of these areas impact or are impacted by the use of measurement.

Software Measures and Measures for Secure Development

This section covers the following three topics:

1. Software Engineering Measurement Process
2. Process Measures for Secure Development
3. Product Measures for Secure Development

Software Engineering Measurement Process

Recent work to establish a common perspective on how to perform software measurement and analysis can be found in International Organization for Standardization and International Electrotechnical Commission (ISO/IEC) 15939 (Software Measurement Process standard), the Capability Maturity Model[18] Integration (CMMI) Measurement and Analysis process area, and the guidance provided by the Practical Software Measurement (PSM) project. For purposes of description, the practices from the CMMI model are presented here. The practices are organized around two major goals or themes: aligning measurement and analysis activities with organizational and project goals and then performing the measurement and analysis activities. Briefly, the practices for aligning measurement are

- Establish Measurement Objectives
- Specify Measures
- Specify Data Collection and Storage Procedures
- Specify Analysis Procedures

The practices for performing measurement are

- Collect Measurement Data
- Analyze Measurement Data
- Store Data and Results
- Communicate Results

These practices, shown as steps in Table 1, are important in that they call for the organization and project to plan their measurement activities so that the right measures are collected, analyzed, and communicated to the appropriate people in an informative format and timely manner. Project management and insight into product quality depend on data that is relevant, reliable, current, and valid. Following these practices (or steps) focuses the measurement activities on the collection of data that will be used, rather than simply collecting data for the sake of measurement. With respect to the development

18. Capability Maturity Model is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

of secure software, it is important that security concerns be addressed in the steps of the measurement process as outlined below.

Table 1. Measurement and Analysis Process

Step Number	Step Name	Input	Techniques	Critical Participants	Output
1	Establish Measurement Objective	System requirements	See Requirements Elicitation practice area	Stakeholders, requirements team	Agreed-to measurement objectives (for which the attainment can be measured)
2	Specify Measures	Measurement objectives, software development life cycle (SDLC)	Facilitated work sessions	Measurement analysts, process engineers, security subject matter experts, users/customers	Measurement definitions for security; focus on problem prone modules; known vulnerabilities; define needed security levels
3	Specify Data Collection and Storage Procedures	Measurement definitions, SDLC	Procedure /process mapping (and potentially design if a current void)	Process engineers, designers, practitioners	Process changes, training needs, tool needs
4	Specify Analysis Procedures	Measurement objectives and definitions (GQM)	Literature review, elicitation	Process engineers, measurement analysts, security experts	Identified statistical and/or qualitative analytical techniques
5	Collect Measurement Data	Measurement plan, data collection tools and infrastructure, instrumented processes	Automated tools and manual forms associated with artifact inspections and testing	Practitioners, testers, measurement analysts, quality assurance	Data in usable form (e.g., database, spreadsheet)
6	Analyze Measurement Data	Output of Step 5 (outputs from Figure 1)	Specified in Step 4	Measurement analysts	Summary, graphical displays, detailed results
7	Store Data and Results	Outputs of Steps 5 & 6	Inspection database or test results database	Measurement analysts, database administrators	Retrievable source data and analytical results

8	Communicate Results	Analyst summary, graphical report	Formatted results with interpretation and recommendations	Project engineers, project/line management, security experts	Feedback to development team, program manager

Effective use of the above process relies first on the agreement of measurement objectives, which can be applied to both product and the development process. These goals rely on explicit system requirements—which mean that both functionality and security aspects must be specified early. The organization should assess the risk environment to address probable threats and translate these concerns into specific security requirements as well as design and implement a development process that will satisfy the security requirements.

Following the specification of security related requirements, measurement objectives may be formulated that will provide insight into the satisfaction of the security requirements. Examples of measurement objectives include the following:

- What vulnerabilities have been detected in our products? Are our current development practices adequate to prevent the recurrence of the vulnerabilities?
- What process points are most vulnerable to the introduction of security-related risks (e.g., injecting reused code/modules into programs—where the variables could go unchecked, etc.)?
- What proportion of defects relate to security concerns and requirements? Do defect classification schemes include security categories?
- To what extent do practitioners comply with security-related processes and procedures?
- To what extent are security concerns addressed in the intermediate workproducts (requirements, design, etc.)? Have measures associated with security requirements and their implementation been defined and planned?
- What are the critical and vulnerable modules? Have vulnerabilities been identified and addressed?

Threat modeling (See Threat Modeling module), or the attempt to identify likely types/sources of attack, can also form a significant guiding requirement to the development processes. A recent thesis by Stuart E. Schechter at Harvard's Department of Computer Science attempts to utilize economic models for valuing the discovery of vulnerabilities during development [Schechter 04]. His measurement of security strength depends most on threat scenarios to assign values to vulnerabilities in an effort to extend a market approach to the development process. Many risk and threat methodologies are available publicly, and Microsoft has published extensive materials that delineate the company's approach to analyzing and mitigating threat risks during the SDLC [Microsoft 03, MSDN 04].

Process Measures for Secure Development

Security measurement objectives for the development process should address

- the existence of security policies applicable to the SDLC (roles, procedures, responsibilities. management, coding rules, acceptance/release criteria, etc.)
- compliance to the above
- efficiency and effectiveness over time

It should be noted that the security measurement objectives for the development process are identical to general measurement objectives—they are broken out here to stress the need to include security concerns in process implementation. Such measures could be implemented as part of an organization's quality assurance function. Although targeted for systems development and risk assessment as a whole, useful guidance for measurement of this type can be found in the NIST publication *Security Metrics Guide for Information Technology Systems* [Swanson 03].

Defect density is a commonly used measure of system quality. It is often computed as the number of defects discovered during system test or during the first six months of operational use divided by the size of the system. Estimates of defects remaining in the product (calculated by phase containment, defect depletion, or capture-recapture techniques) form a natural analogue to estimate remaining security vulnerabilities in the software. Phase containment of defects refers to an analytical technique that measures the proportion of defects originating in a phase that are detected within that same phase. It provides a good characterization of the ability of the development process to maintain quality throughout the SDLC. The *INFOSEC Assurance Capability Maturity Model (IA-CMM)* recognizes the impact of quality control by listing "Establishing Measurable Quality Goals" as one of two features that comprise a level 4 rating of Quantitatively Controlled [NSA 04].

Product Measures for Secure Development

In the product context, security measurement objectives may take the form of

- security requirements, which are based on risks determined by threat assessments, privacy policies, legal implications, etc. and can be specified as to extent and completeness
- architecture security, which addresses the specified security requirements
- secure design criteria, where security requirements can be traced
- secure coding practices, where integrity can be assessed and measured

Not all measures need to be complicated. For instance, in the requirement phase it is useful to know whether security-related requirements have been considered for inclusion in the system requirements. This could be measured initially as yes or no. As experience with the measure accrues over time, the measure could evolve to characterize the extent to which security related requirements have been included, perhaps against a standard. Security measurement objectives during the design and coding phases will make use of tools and inspections/reviews. Much of the inspection measurements will be in the form of traditional defect identification checklists, to which security-oriented items can be added. Table 2 lists some sources of vulnerabilities or concerns that have been widely documented, along with a reference to the part of ISO/IEC 9126 [ISO/IEC 03b, c] that has defined a relevant measure. Software inspection checklists could be extended to include review of the issues in the table.

For instance, one could track the percentage of sources of input that have validation checks and associated error handling. That is, checking each input source for length, format, type, etc. and then its associated exit flows—either accepted then executed or as an error/exception and not executed. The target for this measure would be 100%. Note that, while an improvement over no measurement for this type of vulnerability, this simple measure does not address the potentially complex issue of determining the effectiveness of an input validation technique as implemented and whether it should be counted in the tally. This would require ongoing tracking of this measure's performance to characterize the effectiveness of the input validation techniques used.

Table 2. General Code Integrity Issues

• access control   • access controllability (ISO 9126-3)   • access auditability (ISO 9126-3)

- input validation – particularly to address buffer overflows, format string attacks, SQL injection, etc.
- exception handling/error traps (log bad entries, no execute)
- resource management - consumption, retention, race conditions, closure, etc.
- privileges management – principle of least privilege
- system calls, process forks, etc.
- unexpected behavior or system response
- data security issues
  - data security levels (proprietary, classified, personal, etc.)
  - data encryption (ISO 9126-3)
  - data corruption prevention (ISO 9126-3)
- garbage handling
- risk analysis (identified risks, ranked, with impact analysis, and mitigation and fallback plans)
- implementation flaws

Web Applications

- scripting issues
- sources of input
- forms, text boxes, dialog windows, etc.
- regular expression checks
- header integrity
- session handling
- cookies
- framework vulnerabities (java, .net, etc.)
- access control: front door, back door vulnerability assessment
- penetration attempts versus failures
- depth of successful penetrations before detection

Simple measures of enumeration and appropriate security handling for vulnerabilities would provide insight into the security status of the system during development. In addition to the above table, a useful list of “Measurable Security Entities” and “Measurable Concepts” has been published by Practical Software and Systems Measurement [PSM 05].

In addition to inspection techniques, new tools exist for checking design and code for security vulnerabilities and output measurements as results. Although many companies delineate the conceptual basis for their tools, few offer specific guidance regarding the measurements employed.[47] Two

47. Many of these tools are addressed in the Tools category of Build Security In.

companies provide an exception with their use of new measurements not previously found in the literature.

1. Microsoft's *Secure Windows Initiative* uses a new measure—the *Relative Attack Surface Quotient* (RASQ) as initially presented by Michael Howard [Howard 03]. This calculated number is put forth as a cyclomatic complexity measure for security that yields a *relative* metric of a product's "attackability." The measure is based on the identification of all the external exposures in the product code, with the goal of reducing the product's attack profile. It is of limited use, since the measures are meaningful only for like products, but an independent evaluation did confirm the measure's effectiveness [Ernst & Young 03]. Manadhata and Wing from Carnegie Mellon's Computer Science Department also successfully applied the measurement to Linux [Manadhata 04].
2. Ounce Labs' *Prexis* performs contextual analysis of code for vulnerability identification, much like many other static analysis products. Prexis differs by computing a *V-density* measure to relate the number and criticality of vulnerabilities in the code for project decision-making. As an integrated software risk management and vulnerability assessment product, Prexis includes (1) Prexis/Engine: Source Code Vulnerability Scanning and Knowledgebase Core, (2) Management Risk Dashboard, and (3) Developer Remediation Workbench for the product development life cycle [Ounce Labs 05].

Tools

See the above discussion and the BSI modules Black Box Testing Tools, Code Analysis Tools, and Modeling Tools. Note that spreadsheet programs, statistical packages, and database programs can be very helpful for some measurement and analysis purposes. Some vendors also offer tools that harvest data from other databases and repositories to produce a variety of measurement reports.

Various development tools now include static and dynamic capabilities for analyses of security characteristics within the code. Many of these tools continue to perform as black box tests, where the code is built and then submitted to the tool and results are produced as output. White box testing tools have recently become available, which integrate into the development environment, offering interactive feedback and remediation to the developer during the coding process.

Maturity of Practice

Software measurement is becoming a somewhat mature field, as evidenced by professional and international standards, specialized conferences, and several decades of literature and research. In spite of this history, the practice of software measurement is still highly variable among software development organizations, with many doing little to measure their projects and products during development. Very few organizations employ any form of measurement to assess the security characteristics of their products in a quantitative manner during development. Indeed, few even address security concerns in any manner. Very little exists in the published literature concerning the use of software measurement with respect to characterizing security concerns during software development.

Security-Specific Bibliography

Anderson, Ross J. *Security Engineering: A Guide to Building Dependable Distributed Systems*. New York, NY: Wiley, 2001 (ISBN 0471389226).

Application Security, Inc. *Database Security: A Key Component of Application Security*. New York, NY: Application Security, Inc., 2004. http://www.appsecinc.com.

Biszick-Lockwood, Bar. *IEEE P1074 - Standard for Developing Project Life Cycle Processes*. *QualityIT*, July 2005.
http://www.qualityit.net/Resources/WhitePapers/IEEEP1074-2005-RoadmapForOptimizingSecurityInTheSystemAnd

Brown, Keith. *The .NET Developer's Guide to Windows Security*. Boston, MA: Addison Wesley Professional, Microsoft .NET Development Series, 2004 (ISBN 0321228359).

Cannon, J. C. *Privacy: What Developers and IT Professionals Should Know*. Boston, MA: Addison-Wesley Professional, 2004 (ISBN 0321224094).

Corporate Information Security Working Group. *Report of the Best Practices and Metrics Teams*. Subcommittee on Technology and Information Policy, Intergovernmental Relations and the Census, Government Reform Committee, U.S. House of Representatives (Rev. Jan. 10, 2005).

DISAnet. *DoD Information Technology Security Certification & Accreditation Process*. Dec. 20, 2000. http://iase.disa.mil/ditscap/ditsprimer.ppt.

Woody, Carol; Hall, Anthony; & Clark, John. *Can Secure Systems be Built Using Today's Development Processes?* Panel presentation at the European SEPG in London, England, June 17, 2004. http://www.cert.org/archive/pdf/eursepg04.pdf.

Foundstone. *Hacme Bank*™ v1.0, released 9/8/2004 by Foundstone, Inc. http://www.foundstone.com.

Foundstone. *Validator.NET*™, released 3/08/2005 by Foundstone, Inc. http://www.foundstone.com.

Geer, Dan; Soohoo, K.; & Jaquith, A. "Information Security: Why the Future Belongs to the Quants." *IEEE Security and Privacy Magazine 1,* 4 (July-August 2003): 24-32.

Germanow, Abner; Wysopal, Chris; Geer, Dan; & Darby, Chris. *The Injustice of Insecure Software,* @stake Security Briefing, February 2002. http://www.atstake.com/research/reports/acrobat/atstake_injustice.pdf.

Gilliam, D.; Kelly, J.; Powell, J.; & Bishop, M. "Development of a Software Security Assessment Instrument to Reduce Software Security Risk," 144-149. *Proceedings of the 10th IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*. Cambridge, MA, June 20-22, 2001. Los Alamitos, CA: IEEE Computer Society, 2001.

Graf, Kenneth. *Addressing Challenges in Application Security*, A WatchFire whitepaper, 2005. http://www.watchfire.com/resources/Addressing-Challenges-in-App-Security.pdf (2005).

Graff, Mark G. & Van Wyk, Kenneth R. *Secure Coding: Principles and Practices*. Cambridge, MA: O'Reilly, 2003 (ISBN 0596002424).

Grance, Tim; Hash, Joan; & Stevens, Marc. *Security Considerations in the Information System Development Life Cycle; Recommendations of the National Institute of Standards and Technology*, NIST SPECIAL PUBLICATION 800-64 REV. 1,Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD, (June 2004), U.S. Department of Commerce, Technology Administration, National Institute of Standards and Technology,http://csrc.nist.gov/publications/nistpubs/800-64/NIST-SP800-64.pdf (2004).

Hoglund, Greg & McGraw, Gary. *Exploiting Software: How to Break Code.* Boston, MA: Addison-Wesley, 2004 (ISBN 0201786958).

Howard, Michael. *Fending Off Future Attacks by Reducing Attack Surface*. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dncode/html/secure02132003.asp (2003).

Howard, Michael & LeBlanc, David C. *Writing Secure Code, 2nd ed. Redmond, WA:* Microsoft Press, 2002 (ISBN 0735617228).

National Security Agency. *INFOSEC Assurance Capability Maturity Model (IA-CMM), Version 3.1*, Infosec Assurance Training and Rating Program.

http://www.iatrp.com/IA-CMMv3_1-%20FINAL-NOV04.doc[69] (2004).

Institute for Security and Open Methodologies (ISECOM). *SPSMM - The Secure Programming Standards Methodology Manual*. http://isecom.securenetltd.com/spsmm.0.5.1.en.pdf (2001).

Jaquith, Andrew. *The Security of Applications: Not All Are Created Equal*, @atstake Security Research Report. http://www.atstake.com/research/reports/acrobat/atstake_app_unequal.pdf (2002).

Kimbell, John & Walrath Marjorie. "Life Cycle Security and DITSCAP" *IA Newsletter*, vol. 4, no. 2, Spring 01, http://iase.disa.mil/ditscap/ditsarticle.pdf (2001).

Koziol, Jack; Litchfield, D.; Aitel, D.; Anley, C.; Eren, S.; Mehta, N.; & Riley. H. *The Shellcoder's Handbook: Discovering and Exploiting Security Holes*. Indianapolis, IN: Wiley Pub, 2004 (ISBN 0764544683).

Letteer, Ray A. *Information Operations and the DAA (Designated Approving Authority)*, DISA/IPMO D253 [SAIC]. http://iase.disa.mil/ditscap/daav3.ppt (2001).

Levine, Matthew. *The Importance of Application Security*. @atstake Security Briefing, January 2003. http://www.atstake.com/research/reports/acrobat/atstake_application_security.pdf (2003).

Manadhata, Pratyusa & Wing, Jeannette M. *Measuring a System's Attack Surface CMU-CS-04-102*. Pittsburgh, PA: School of Computer Science, Carnegie Mellon University, January 2004.

Mead, Nancy R. *International Liability Issues for Software Quality* (CMU/SEI-2003-SR-001, ADA416434). Pittsburgh, PA: CERT Research Center, Software Engineering Institute, Carnegie Mellon University, July 2003.

Microsoft TechNet. *Threats and Countermeasures Guide*. http:microsoft.com/technet/security/topics/Serversecurity/tcg/tcgch00.mspx (2003).

Microsoft Developer Network (MSDN). *Threat Modeling: Patterns and Practices*. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/secmod/html/secmod76.asp?frame=true&hidetoc=tru
(2004).

National Institute of Standards and Technology. *Revised NIST SP 800-26* (*Security Self-Assessment Guide for Information Technology Systems*, November 2001) *System Questionnaire with NIST SP800-53* (*Recommended Security Controls for Federal Information Systems*, February 2005 (Including updates through 04-22-2005)) *References and Associated Security Control Mappings*. http://csrc.nist.gov/publications/nistpubs/800-26/Mapping-of-800-53v1.doc (2005).

National Institute of Standards and Technology. *Information Security in the System Development Life Cycle (SDLC) Brochure*. http://csrc.nist.gov/SDLCinfosec/SDLC_brochure_Aug04.pdf (2004).

Ounce Labs. *Product Overview*. http://www.ouncelabs.com/overview.html (2005).

OWASP. *A Guide to Building Secure Web Applications and Web Services, 2.0 Black Hat Edition*. The Open Web Application Security Project (OWASP). http://easynews.dl.sourceforge.net/sourceforge/owasp/OWASPGuide2.0.1.pdf (2005).

Peikari, Cyrus & Chuvakin, Anton. *Security Warrior*. Sebastopol, CA :O'Reilly & Associates, Inc, 2004 (ISBN 0596005458).

President's Information Technology Advisory Committee (PITAC), *Cyber Security: A Crisis of Prioritization*, National Coordination Office for Information Technology Research and Development, Arlington, VA. http://www.nitrd.gov/pitac/reports/20050301_cybersecurity/cybersecurity.pdf (2005).

69. http://www.iatrp.com/IA-CMMv3_1-%2520FINAL-NOV04.doc

Practical Software and Systems Measurement (PSM). *Security Measurement, White Paper v2.0*, http://www.psmsc.com/Downloads/Other/Security%20White%20Paper%202.0.pdf[81] (2005).

Ross, Ron; Swanson, Marianne; Stoneburner, Gary; Katzke, Stu; & Johnson, Arnold. *Guide for the Security Certification and Accreditation of Federal Information Systems*, NIST Special Publication 800-37, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD. U.S. Department of Commerce, Technology Administration, National Institute of Standards and Technology http://csrc.nist.gov/publications/nistpubs/800-37/SP800-37-final.pdf (2004).

Sademies, Anni. *Process Approach to Information Security Metrics in Finnish Industry and State Institutions*, VTT Electronics, Oulu, Finland, http://www.vtt.fi/inf/pdf/publications/2004/P544.pdf (2004).

Schechter, Stuart Edward. *Computer Security Strength & Risk: A Quantitative Approach*, Doctoral Thesis, Computer Science, Harvard University, Cambridge, Massachusetts, May 2004. http://www.eecs.harvard.edu/~stuart/papers/thesis.pdf[84] (2004).

Seacord, Robert C. *Secure Coding in C and C++*. Boston, MA: Addison-Wesley, 2005 (ISBN 0321335724).

SecureSoftware, Inc. *A Special Report for managers: Why Application Security Is the New Business Imperative – and How to Achieve It.*, Secure Software, Inc., McLean VA. http://modeldrivenengineering.org/pub/Sandbox/ProcessesForSecurity/appsec.pdf (2004).

Soo Hoo, Kevin; Jaquith, Andrew; & Geer, Dan. *The Security of Applications, Reloaded*, @atstake Security Briefing, July, 2003. http://www.atstake.com/research/reports/acrobat/atstake_app_reloaded.pdf (2003).

Stoneburner, Gary; Hayden, Clark; & Feringa, Alexis. *Engineering Principles for Information Technology Security (A Baseline for Achieving Security), Revision A*, NIST Special Publication 800-27 Rev A, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD, and Booz-Allen and Hamilton U.S. Department of Commerce, Technology Administration, National Institute of Standards and Technology. http://csrc.nist.gov/publications/nistpubs/800-27A/SP800-27-RevA.pdf (2004).

Swanson, Marianne; Wohl, Amy; Pope, Lucinda; Grance, Tim; Hash, Joan; & Thomas, Ray. *Contingency Planning Guide for Information Technology Systems; Recommendations of the National Institute of Standards and Technology*, NIST Special Publication 800-34. U.S. Department of Commerce, Technology Administration, National Institute of Standards and Technology. http://csrc.nist.gov/publications/nistpubs/800-34/sp800-34.pdf (2002).

Swanson, Marianne; Bartol, Nadya; Sabato, John; Hash, Joan; & Graffo, Laurie. *Security Metrics Guide for Information Technology Systems*, NIST Special Publication 800-55, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD, U.S. Department of Commerce, Technology Administration, National Institute of Standards and Technology. http://csrc.nist.gov/publications/nistpubs/800-55/sp800-55.pdf (2003).

Trocino, Douglas P. *Developing Secure Applications: Best Practices for Writing Secure Code*. Boca Raton, Fla.: Auerbach , 2004 (ISBN 0849319900).

Viega, John & McGraw, Gary. *Building Secure Software: How to Avoid Security Problems the Right Way*. Boston, MA: Addison-Wesley, 2002 (ISBN 020172152X).

81. http://www.psmsc.com/Downloads/Other/Security%2520White%2520Paper%25202.0.pdf

84. http://www.eecs.harvard.edu/%7Estuart/papers/thesis.pdf

Viega, John & Messier, Matt. *Secure Programming Cookbook for C and C++*. Sebastopol, CA : O'Reilly, 2003 (ISBN 0596003943).

Wheeler, David A., *Secure Programming for Linux and UNIX HOWTO*, v3.010, (March 3, 2003). http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/pdf/Secure-Programs-HOWTO.pdf (2003).

Whittaker, James A. & Thompson, Herbert H. *How to Break Software Security: Effective Techniques for Security Testing*. Boston: Pearson/Addison Wesley, 2004 (ISBN 0321194330).

IBM Developer Works Series:

Wheeler, David A. Secure programmer: *Call components safely*. http://www-106.ibm.com/developerworks/linux/library/l-calls.html (December 16, 2004).

Wheeler, David A. Secure programmer: *Prevent race conditions*. http://www-106.ibm.com/developerworks/linux/library/l-sprace.html (October 7, 2004).

Wheeler, David A. Secure programmer: *Minimizing privileges*. http://www-106.ibm.com/developerworks/linux/library/l-sppriv.html (May 20, 2004).

Wheeler, David A. Secure programmer: *Countering buffer overflows* http://www-106.ibm.com/developerworks/linux/library/l-sp4.html (January 27, 2004).

Wheeler, David A. Secure programmer: *Keep an eye on inputs* http://www-106.ibm.com/developerworks/linux/library/l-sp3.html (December 19, 2003).

Wheeler, David A. Secure programmer: *Validating input* http://www-106.ibm.com/developerworks/linux/library/l-sp2.html (October 23, 2003).

Wheeler, David A. Secure programmer: *Developing secure prog*rams http://www-106.ibm.com/developerworks/linux/library/l-sp1.html (August 21, 2003).

Software Engineering Bibliography

Beizer, Boris. *Software Testing Techniques, 2nd edition*, Boston, MA.: International Thomson Computer Press, 1990 (ISBN 1850328803).

Chrissis, M. B.; Konrad, M.; & Shrum, S. *CMMI: Guidelines for Process Integration and Product Improvement*. Boston, MA: Addison-Wesley, 2003 (ISBN 0321154967).

C?T?, Marc-Alexis; Suryn, Witold; Martin, Robert A.; & Laporte, Claude Y. "Evolving a Corporate Software Quality Assessment Exercise: A Migration Path to ISO/IEC 9126." *Software Quality Engineering 6*, 3. http://www.asq.org/pub/sqp/past/vol6_issue3/SQPv6i3cote.pdf (2004).

Ernst & Young LLP. *Using Attack Surface Area And Relative Attack Surface Quotient To Identify Attackability*, Security & Technology Solutions**,** Advanced Security Center. Customer Information Paper, 2003. http://www.microsoft.com/windowsserver2003/docs/AdvSec.pdf (2003).

Fenton, Norman E. & Pfleeger, Sharon L. *Software Metrics: A Rigorous and Practical Approach*, 2nd ed. Boston, MA: International Thomson Computer Press, 1996 (ISBN 1850322759).

Grady, Robert B. *Practical Software Metrics for Project Management and Process Improvement*. Englewood Cliffs, NJ : Prentice Hall, 1992 (ISBN 0137203845).

Halstead, Maurice.H. *Elements of Software Science*. New York, NY: Elsevier, 1977 (ISBN 0444002057).

Humphrey, Watts S. *Managing the Software Process*. Reading, MA: Addison-Wesley, 1989 (ISBN 0201180952).

Humphrey, Watts S. *A Discipline for Software Engineering*. Reading, MA: Addison-Wesley, 1995 (ISBN 0201546108).

Humphrey, Watts S. *Introduction to the Team Software Process*. Reading, MA: Addison-Wesley, 2000 (ISBN 020147719X).

ISO. *ISO/IEC 15939:2002, Software engineering – Software Measurement Process*. Geneva, Switzerland: International Organization for Standardization, 2002.

ISO. *ISO/IEC 9126-1:2001: Software Engineering – Product Quality. Part 1: Quality Model*. Geneva, Switzerland: International Organization for Standardization, 2001.

ISO. *ISO/IEC 9126-2:2003: Software Engineering – Product Quality. Part 2: External Metrics*. Geneva, Switzerland: International Organization for Standardization, 2003.

ISO. *ISO/IEC 9126-2:2003: Software Engineering – Product Quality. Part 3: Internal Metrics*. Geneva, Switzerland: International Organization for Standardization, 2003.

ISO. *ISO/IEC 9126-2:2004: Software Engineering – Product Quality. Part 2: Quality in Use Metrics*. Geneva, Switzerland: International Organization for Standardization, 2004.

Kan, Stephen H. *Metrics and Models in Software Quality Engineering*, 2nd ed. Boston, MA: Addison-Wesley, 2003 (ISBN 0201729156).

McGarry, John; Card, David; Jones, Cheryl; Layman, Beth; Clark, Elizabeth; Dean, Joseph; & Hall, Fred. *Practice Software Measurement: Objective Information for Decision Makers*, Boston, MA: Addison-Wesley, 2002 (ISBN 0201715163).

Reports and Articles

Basili, Victor R. "Quantitative Software Complexity Models: A Panel Summary." *IEEE Proceedings of the Workshop on Quantitative Software Models for Reliability, Complexity, and Cost*. October 1979. http://www.cs.umd.edu/projects/SoftEng/ESEG/papers/83.14.pdf (1979).

Basili, Victor R. & Weiss, David M. "A Methodology for Collecting Valid Software Engineering Data." *IEEE Transactions on Software Engineering 10*, 6 (November 1984): 728-738.

Fagan, Michael E. "Design and code inspections to reduce errors in program development." *IBM Systems Journal 38*, 2 & 3 (1999): 258-287.

Florac, W. Software Quality Measurement: A Framework for Counting Problems and Defects (CMU/SEI-92-TR-022, ADA258556). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1992. http://www.sei.cmu.edu/publications/documents/92.reports/92.tr.022.html.

Fenton, Norman. E. & Ohlsson, Niclas. "Quantitative Analysis of Faults and Failures in a Complex Software System." *IEEE Transactions on Software Engineering 26*, 8 (August 2000): 797-814.

McCabe, T. "A Complexity Measure." *IEEE Transactions on Software Engineering 2*, 4 (December 1976): 308-320.

McGraw, Gary. "Software Security." *IEEE Security and Privacy 2*, 2 (March-April 2004): 80-83.

Web Articles/Artifacts/Tools

Foundstone, Inc. *Hacme Bank™ v1.0* (released 9/8/2004). http://www.foundstone.com/index.htm?subnav=resources/navigation.htm&subcontent=/resources/proddesc/hacmebar

(2004).

Practical Software and Systems Measurement. http://www.psmsc.com (2005).

Software Engineering Information Repository. http://seir.sei.cmu.edu (2005).

Capture/Recapture Analysis

Briand, Lionel C.; Emam, Khaled El; Freimut, Bernd G.; & Laitenberger, Oliver. “A Comprehensive Evaluation of Capture-Recapture Models for Estimating Software Defect Content.” *IEEE Transactions on Software Engineering 26*, 6 (June 2000): 518-540.

Humphrey, Watts S. *Introduction to the Team Software Process*. Reading, MA: Addison Wesley, 1999 (ISBN 020147719X).

Petersson, Hakan & Wohlin, Claes. “An Empirical Study of Experience-Based Software Defect Content Estimation Methods,” 126-135. *Proceedings of the International Symposium on Software Reliability Engineering, ISSRE*. Boca Raton, FL, Nov. 1-4, 1999. Los Alamitos, CA: IEEE Computer Society, 1999.

Defect Prevention Program

Mays, R. G.; Jones, C. L.; Holloway, G. J.; & Studinski, D. P. “Experiences with Defect Prevention.” *IBM Systems Journal 29*, 1 (1990): 4-32.

Grady, R. B. “Software Failure Analysis for High-Return Process Improvement Decisions.” *Hewlett Packard Journal 47, 4* (August 1996): 15-24.

Gale, J. L.; Tirso, J. R.; & Burchfield, C. A. “Implement the Defect Prevention Process in the MVS Interactive Programming Organization.” *IBM Systems Journal 29*, 1 (1990): 33-43.

Statistical Process Control

Florac, William. A. & Carleton, Anita D. *Measuring the Software Process: Statistical Process Control for Software Process Improvement*. Reading, MA: Addison Wesley, 1999 (ISBN 0201604442).

Orthogonal Defect Classification Defect Prediction Technique

Chillarege, Ram; Bhandari, Inderpal S.; Chaar, Jarir K.;Halliday, Michael J.; Moebus, Diane S.; Ray, Bonnie K.; & Wong, Man-Yuen. “Orthogonal Defect Classification - A Concept for In-Process Measurements.” *IEEE Transactions on Software Engineering 18*, 11 (Nov. 1992): 943-956.

Bridge, Norman & Miller, Corrine. “Orthogonal Defect Classification: Using Defect Data to Improve Software Development,” 197-213. *International Conference on Software Quality*. Montgomery, AL, October 6-8, 1997. Milwaukee, WI: American Society for Quality, 1997.

El Emam, K. & Wieczorek, I. “The Repeatability of Code Defect Classifications,” 322-333. *Proceedings of the Ninth International Symposium onSoftware Reliability Engineering*. Paderborn, Germany, Nov. 4-7, 1998. Los Alamitos, CA: IEEE Computer Society, 1998.

Fault Proneness

Selby, R. & Basili, V. “Analyzing Error-Prone System Structure.” *IEEE Transactions on Software Engineering 17*, 2 (Feb. 1991): 141-152.

Briand, Lionel C.; Melo, Walcelio L.; & Wust, Jurgen. “Assessing the Applicability of Fault-Proneness Models Across Object-Oriented Software Projects.” IEEE *Transactions on Software Engineering 28*, 7 (July 2002): 706-720.

El Emam, K. “A Primer on Object Oriented Measurement,” 185-187. *7th International Software Metrics Symposium*. London, England, April 4-6, 2001. Los Alamitos, CA: IEEE Computer Society, 2001.

Fenton, Norman E. & Ohlsson, Niclas. “Quantitative Analysis of Faults and Failures in a Complex Software System.” *IEEE Transactions on Software Engineering 26*, 8 (August 2000): 797-814.

Ohlsson, Magnus C. & Wohlin, Claes. “Identification of Green, Yellow, and Red Legacy Components,” 6-15. *Proceedings of the 1998 IEEE International Conference on Software Maintenance, ICSM*. Bethesda, MD, Nov. 16-20, 1998. Los Alamitos, CA: IEEE Computer Society, 1998.

General Defect Detection References

Fenton, Norman E. & Neil, Martin. “A Critique of Software Defect Prediction Models.” *IEEE Transactions on Software Engineering 25*, 5 (Sept. 1999): 675-689.

Frederick, M. “Using Defect Tracking and Analysis to Improve Software Quality.” University of Maryland. http://www.dacs.dtic.mil/techs/defect/defect.pdf (1999).

Florac, W. A. *Software Quality Measurement: A Framework for CountingProblems and Defects* (CMU/SEI-92-TR-22, ADA258556). Pittsburgh PA: Software Engineering Institute, Carnegie Mellon University, September 1992.

Peng, Wendy W. & Wallace, Dolores R. *Software Error Analysis*. Gaithersburg, MD: U.S. Dept. of Commerce, National Institute of Standards and Technology, 1993.

Empirical Defect Prediction

Humphrey, W. *Introduction to the Team Software Process*. Reading, MA: Addison Wesley, 2000 (ISBN 020147719X).

Weller, E. F. “Using metrics to manage software projects.” *IEEE Software 27*, 9 (Sept. 1994): 27-33.

Defect Profile Prediction Technique

Gaffney, John; Roberts, William; & DiLorio, Robert. “A Process and Tool for Improved Software Defect Analysis and Quality Management,” Track 7, 463-469. *CD-ROM Proceedings for the Ninth Annual Software Technology Conference: Information Dominance Through Software Technology*. Salt Lake City, Utah, April 27 – May 2, 1997. Hill AFB, UT: Software Technology Support Center (STSC), 1997.

COQUALMO Prediction Technique

Chulani, Sunita & Boehm, Barry. *Modeling Software Defect Introduction and Removal: COQUALMO* (Technical Report USC-CSE-99-510). Los Angeles, CA: University of Southern California, Center for Software Engineering, 1999.
http://sunset.usc.edu/publications/TECHRPTS/1999/usccse99-510/usccse99-510.pdf[109] (1999).

SEI Copyright

109. http://sunset.usc.edu/publications/TECHRPTS/1999/usccse99-510/usccse99-510.pd

Fields

Name	Value
Copyright Holder	SEI

Fields

Name	Value
is-content-area-overview	true
Content Areas	Best Practices/Measurement
SDLC Relevance	Implementation Maintenance
Workflow State	Publishable

1. http://www.sei.cmu.edu/about/legal-permissions.html